METHOD FOR EMULATING MULTI-PROCESSOR ENVIRONMENT

CROSS REFERENCE TO CO-PENDING APPLICATIONS

The present application is related to co-pending Patent Application Serial No.

5   33012/283/101, filed _____, entitled "A Method of Dispatching Target Instructions Using

C Code", assigned to the assignee of the present invention and incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention - The present invention generally relates to digital data processing

10   systems, and more particularly relates to emulation of a first device through the use of a second

and different type of device.

2. Description of the Prior Art - It is well known in the prior art to design and build general

purpose instruction processors having a predefined software architecture and instruction set. It is

also known to design and build instruction processors having predefined software architectures

15   and instruction sets which are more specialized in nature. Such specialized designs are typically

directed toward optimization of a particular task or a closely related group of tasks.

One of the tasks to which a specialized design is particularly pertinent is in the field of

real time communications processing. Whereas this application does not require complex

arithmetic and mathematical calculations, for example, it does require rapid switching of data

20   and control. It also relies upon rapid transfer of data from one location to another. As a result, a

communications instruction processor typically has extensive switching, timing, and data transfer

instruction, with only modest attention to complex mathematics, such as floating point

arithmetic. In a typical implementation, this means that the communications processor would

1

have a modest set of general purpose instructions (e.g., add, shift, compare, etc.) along with a number of specialized instructions for queue manipulation, segment handling, buffer allocation, etc.

Perhaps the best example of a communications instruction processor having an especially effective design is the Distributed Communications Processor (DCP) available from Unisys Corporation. The DCP instruction processor is found in many systems in use today. The unique software architecture which provides the efficiencies for this specialized application provides difficulty in upgrading to newer and faster hardware.

It is normal to expect that newer and faster instruction processors are continuously being designed and built. Yet for economic reasons of scale, these designs typically have a more generalized software architecture to promote a larger volume of applications over which to amortize the non-recurring development costs. Thus, to benefit from these newer instruction processors, the existing DCP communications software would need to be reprogrammed to operate within the more generalized software architecture of the new instruction processor.

Such reprogramming has two major disadvantages. First, the cost of the reprogramming process may be substantial and occupy a rather extensive development schedule. Second, programming of the communications software to operate within a more generalized architecture does away with the efficiencies to be realized by the more specialized communications architecture.

The typical answer to this dilemma is emulation. Emulation is the process whereby the host machine (in this case a new instruction processor design having a somewhat generalized software architecture) is programmed to provide the more specialized software architecture of the target machine (in this case the DCP). Using this technique, the existing DCP communications

2

software can run on a newer instruction processor with an apparently incompatible software architecture. U.S. Patent No. 5,794,011, issued to Paul et al., discusses the technique of emulation.

Thus, emulation most easily resolves the first of the problems by eliminating the expense and long development schedule involved in reprogramming. However, the question of runtime efficiencies is always present when emulation techniques are employed. Unless the specific emulation system is optimized for efficiency, the new and faster host instruction processor may actually executed the desired software more slowly than the older specialized target processor. The easiest way to provide efficient emulation is to provide the emulation logic in microcode. As such, the host machine becomes the target machine via the microprogramming. Unfortunately, this directly changes the host environment to look like the target environment. As a result, the host machine loses its character and can no longer execute code specifically designed for its native environment.

In order to provide an emulation wherein the host machine presents both the target environment and its native environment, it is common for prior art emulation techniques to perform a runtime software comparison within the host machine to locate those host machine instructions required to implement a particular target instruction. For emulation of the DCP, this would require an average of about 150 comparisons per specialized target instruction to be emulated. This would render a hopelessly inefficient emulation.

Of particular difficulty is the emulation of a multi-processor target environment. This difficulty is exacerbated when utilizing a single processor host.

3

SUMMARY OF THE INVENTION

The present invention overcomes many of the disadvantages found in the prior art by providing a method of and apparatus for software emulation of a multi-processor target on a single processor host machine while continuing to provide the native host environment. Unlike prior art systems which utilize comparisons to select the host instructions required to emulate a particular specialized target instruction, the present invention uses the target instruction as a direct pointer to locate the necessary host instructions.

In accordance with the preferred mode of the present invention, a Pentium microprocessor chip, available from Intel Corporation, is used as the host for emulating the Distributed Communications Processor (DCP), available from Unisys Corporation. The invention utilizes a fast index to the target instruction code used by the processor object. An array of procedures is defined in the application within a header file. Each array is defined as a pointer. The pointer is initially void. The real values are filled in by the constructor of the processor object during the initialization of the processor. Using this array, the processor flow is directed to the single instruction using the op code and an additional four bit field as a direct index to the routine.

When a target instruction is processed by the processor object, it is immediately dispatched using the op code as an index into an array of target instruction pointers. This is an efficient alternative to running a string of comparisons between the op codes and target instructions.

This immediate dispatch of the processor object means that the target instruction is promptly emulated without the time consuming comparison found in the prior art. Thus, the run time execution of the emulator becomes substantially faster and more efficient.

4

A separate object is utilized for each processor to be emulated. Thus, in emulating a multi-processor environment, a plurality of objects are used wherein each object corresponds to a different target processor. Emulation of a multi-processor target environment by a single processor provides significant flexibility in implementing the emulation system. Obviously, this

5   permits the total multi-processor emulation to be accomplished in a single processor system. Furthermore, in multi-processor host systems, the multi-processor target may be emulated from a single or less than all of the host processors which oftentimes gives a much more efficient division of labor amongst processors.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, in which like reference numerals designate like parts throughout the figures thereof and wherein:

~~FIG. 1 is a conceptual diagram of a typical legacy environment employing the DCP.~~

FIG. 2 is a conceptual diagram of modern network architectures;

FIG. 3 is a conceptual diagram of a modern network integrated with the DCP;

FIG. 4 is a diagram showing the basic DCP architecture;

FIG. 5 is a diagram showing an application implementing the DCP system

FIG. 6 is a an illustration of the operation of the DCP;

FIG. 7 is a detailed diagram showing linking of the host and target instructions in accordance with the present invention;

FIG. 8 is detailed diagram showing operation of a multiprocessor system; and

FIG. 9 is a detailed diagram showing I/O operations.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 is a conceptualized diagram showing a legacy system 10 employing the DCP. This basic legacy system is typical of installed data management systems of various enterprises of differing sizes. Channel connect 12 represents the direct coupling of Host 14 to DCP 18 via cable 16. In the normal situation, Channel connect 12 shows that Host 14 and DCP 18 are physically collocated. In this installation, DCP 18 functions as a data concentrator and interface to single function terminals 22 and 24. These single function terminals are not collocated with the other components of legacy system 10 but are coupled via dedicated link 20, as shown. Use of DCP 18 in this way permits modification and upgrade of single function terminals 22 and 24, along with dedicated link 20, without impact on host 14 and its contained legacy software base.

Fig. 2 is a conceptualized diagram 26 of the various components of modern network

architectures. Host A 28 is representative of a typical Unisys Model 2200 host configuration

currently available from Unisys Corporation. Fiber network, FDDI Network 30, couples the

5   various components which are not physically collocated with Host A 28. Remote components 38

and 40, coupled via path 34, may be a combination of single function terminals, "intelligent"

terminals, other computers, data concentrators, servers, etc.

Unix Host 36 is shown as a component providing a different applications environment.

X.25 Network 32 is shown as a component providing a different networking protocol. Novell

10   LAN 44, shows a local area network which couples industry compatible, personal computers 60

and 62. This configuration may be found in the typical small office.

Host B 50 is typically a legacy host available from IBM Corporation. It couples with

SNA Network 52 to Token Ring 54. Multiple personal computers, servers, and concentrators

may be coupled to and serviced by Token Ring 54. TCP/IP Device 42 shows yet another

15   networking protocol in servicing components 56 and 58 via path 46. Components 56 and 58 may

be terminals, computers, servers, etc.

The challenge in integrating these various networks and systems is that which is

addressed by the Distributed Communication Processor (DCP) available from Unisys

Corporation. This challenge is met through the use of an architecture having a hybrid instruction

20   set with both general purpose and special purpose instructions.

Fig. 3 is a conceptualized diagram 62 showing integration of diverse networks and systems through utilization of the DCP technology. The heart of this integrated system is DCP 600 Series 76, as shown. It couples to the remaining system components via the native

5 communication protocol of each. Thus, each of the system components is functionally interconnected without the need to appreciably modify any of these components to accommodate the differing communication protocols utilized by the other system components.

For example, TCP/IP Device 70 communicates with DCP 76 in accordance with its defined protocol. That permits Ethernet LAN devices 78 and 82, coupled to TCP/IP Device 70

10 via Ethernet LAN 80, to communicate with DCP 76 and therefore with any device coupled to DCP 76. Similarly, UNIX Host 64 communicates with DCP 76 according to its communication protocol.

In like manner, Host A 66, typically a Model 2200 system available from Unisys Corporation, is functionally coupled to DCP 76. Thus, devices 72 and 74, coupled to Host A 66

15 via FDDI Network 68, can readily communicate with UNIX Host 64, TCP/IP Device 70, and all devices coupled to DCP 76. In addition, Host B 86, typically and IBM mainframe, along with its SNA Network 88 and Token Ring 90, communicate with DCP 66 in its preferred protocol. And Novell LAN 84, along with LAN devices 94 and 96, communicate with DCP 66 as if it were a Novell LAN device.

Fig. 4 is a block diagram of the basic DCP hardware architecture. Communication

Processor 98 is the primary instruction processor of the DCP system. It is this element which is

emulated in accordance with the present invention as described in detail below.

5   Communication Processor 98 is coupled to Local Storage 102 via bi-directional memory

bus 100, as shown. Similarly, bi-directional memory bus 104 couples Local Storage 102 to

Input/Output Processor 106. Line Modules 118, 120, 122, 124, 126, 128, 130, and 132 provide

the individualized electrical and functional interface to the various diverse networks within the

system (see also Fig. 3). Mass Storage 116 provides the required large storage capacity.

10   Communication lines 108, 110, 112, and 114 couple directly to various diverse components of

the integrated system.

Fig. 5 is a block diagram of the application implementation of the emulated DCP system. In accordance with the preferred mode of practicing the present invention, a COM32 microprocessor, available from Intel Corporation, is utilized as the host processor. The preferred programming language is C++. The applications run under the Windows NT operating system. This configuration permits direct use of such software as Telcon, thus directly providing a number of the diverse system interfaces.

CP Implementation 134 is the application associated with emulation of the DCP architecture. DCP DLL 136 provides the instruction sequences as discussed in detail below. The DCP input/output control is performed by IOP Implementation 142, with PP DLL 144 providing the coding sequences. DCP Line Modules 146, 148, and 150 are thus DCP style interfaces coupling to a number of diverse components employing a variety of network protocol standards.

Winsock or Handler 138 provide the opportunity to interface directly from the Host (i.e., Intel) Native environment under Windows NT control to system components communicating with compatible interfaces. Driver 140 is typically a Windows NT Driver. NIC or TCP/IP 152 couples directly to compatible communication links.

Fig. 6 is a detailed diagram showing the communications processor applications of the preferred emulation technique of the present invention. Initialization 154 is utilized to preset operating conditions. Processor 156 contains Dispatcher 158, Storage Access Methods 160, and Registers 162.

Instructions 170 contains the host instructions required to implement a particular target instruction. Typical instructions include Load 172, Jump 174, Arithmetic 176, Store 178, additional instructions 180, and 300 additional instructions 182. Dispatcher makes the selection as shown and the responses include providing an corresponding instruction to Dispatcher 158 or a requested memory operation via Storage Access Methods 160.

When a memory operation is required, Storage Access Methods 160 requests the operation from Main Storage and Locks 168. Dispatcher 158 preserves its real time processing native environment as shown by Real Time Clock 166 and Events 164. Such indications are communicated to Dispatcher 158 as shown.

Fig. 7 is a detailed diagram showing how the array of procedures 184 points directly to the corresponding instructions 186. The instructions are decoded using the op code and a 4-bit field. These two fields are combined to point directly to the appropriate one of instructions 186.

5   For example, Instr. (N) 188 is decoded to point to Load 190, and Instr. (N+1) 192 is decoded to point to Jump 194. Similarly, Instr. (N+2) is decoded to point to Arithmetic 198, and Instr. (N+3) 200 is decoded to point to Store 202. Additional instructions 204 are decoded to point to their corresponding instructions 206.

Fig. 8 is a detailed diagram showing emulation of a multiprocessor DCP configuration, which incorporates the preferred mode of the present invention. Even in a multiprocessor host configuration, a single host processor may be utilized to emulate multiple target processors. In this embodiment, a plurality of application processors, including Processor 0 214 through Processor n 218, are defined. Each of the processors operates as previously discussed (see also Figs. 6 and 7). Thus, Processor 0 214 has a Dispatcher 220, Storage Access 22, and Registers 224. Similarly, Processor n 218 has a Dispatcher 230, Storage Access 232, and Registers 234. DCP Storage and Storage Locks 212 operates as previously described for single processor emulation. Instructions 216 are as previously described.

In a typical example, Dispatcher 220 of Processor 0 214 combines an op code and corresponding four bit field producing an index to uniquely specify Add Register 240. Access 236 is made to Add Register 240. In response Request 238 to access the necessary data is made to Storage Access 222 which in turn formats Request 226 to DCP Storage and Storage Locks 212.

In a second example, Dispatcher 230 of Processor n 218 combines an op code and corresponding four bit field which uniquely identifies Compare Constant 246. Access 242 transfers the index to Instructions 216 and Compare Constant 246 is accessed. Request 244 is made of Storage Access 232, which in turn formulates Request 228 for accessing DCP Storage and Storage Locks 212.

Fig. 9 is a detailed diagram showing operation of an emulated input/output instruction. This operates similar to emulation of communications processor instructions. IOP Object 250 contains three major components. Dispatcher 256 and Storage Access Methods 252 operate similar to the corresponding components in the communications processor applications (see also Figs. 6 and 7). Sequencer 258 operates to sequence transmissions which are often handled in specialized hardware in the target machine. These operations include sequencing block transmissions, managing serial transmissions, etc. Sequencer 258 drives applications for individual input/output channels including PP Chain 1 278, PP Chain 2 280, and PP Chain n 282 via path 276.

Main Storage and Locks 248 operates as previously described. Instructions 274 are specialized input/output instructions. These include, Queue 264, Start Data 266, Throttle 268, Jump 270, and Move Data 272.

In the present example, Dispatcher 256 combines the present op code and corresponding four bit field to produce an index to uniquely address Start Data 266. The access is made via path 262. In response, Start Data 266 communicates with Storage Access Methods 252 to make the necessary memory accesses via path 260. Storage Access Methods 252 makes the memory accesses via path 254 coupled to Main Storage and Locks 248.

Having thus described the preferred embodiments of the present invention, those of skill in the art will readily appreciate that the teachings found herein may be applied to yet other embodiments within the scope of the claims hereto attached.

5

What is claimed is: